DNA Alignment – Worksheet 3

*CDS 230*

*Spring 2018*

# Problem

What enables the double for loops in the example on slide 5 of the video to improve performance as opposed to regular nested Python loops?

Instead of reading both for loops in python, the python program calls code that has already been compiled. The compiled program is faster than python and it only computes the numbers for the upper side of the diagonal. Only one nested loop is actually in python.

# Problem

Load and run the code presented in slide 10 of the video (Code 25.19 in the text).

import blosum

sq1 = 'KMTIFFMILK'

sq2 = 'NQTIFF'

subvals = dpg. FastSubValues (blosum.BLOSUM50, blosum.PBET, sq1, sq2)

scmat, arrow = dpg. FastSW (subvals, sq1, sq2)

t1, t2 = dpg. SWBacktrace (scmat, arrow, sq1, sq2)

print(subvals, scmat, arrow, t1, t2)

*IQIFSFIFRQ--EWNDA*

*-QIF-FFFRMSVEWND-*

*[[ 0 0 0 0 0 0 0]*

*[ 0 0 2 -1 -3 -4 -4]*

*[ 0 -2 0 -1 2 0 0]*

*[ 0 0 -1 5 -1 -2 -2]*

*[ 0 -3 -3 -1 5 0 0]*

*[ 0 -4 -4 -2 0 8 8]*

*[ 0 -4 -4 -2 0 8 8]*

*[ 0 -2 0 -1 2 0 0]*

*[ 0 -3 -3 -1 5 0 0]*

*[ 0 -4 -2 -1 2 1 1]*

*[ 0 0 2 -1 -3 -4 -4]] [[ 0 0 0 0 0 0 0]*

*[ 0 0 2 0 0 0 0]*

*[ 0 0 0 1 2 0 0]*

*[ 0 0 0 5 0 0 0]*

*[ 0 0 0 0 10 2 0]*

*[ 0 0 0 0 2 18 10]*

*[ 0 0 0 0 0 10 26]*

*[ 0 0 0 0 2 2 18]*

*[ 0 0 0 0 5 2 10]*

*[ 0 0 0 0 2 6 3]*

*[ 0 0 2 0 0 0 2]] [[1 1 1 1 1 1 1]*

*[0 2 2 3 3 3 3]*

*[0 3 2 2 2 2 2]*

*[0 2 3 2 2 2 3]*

*[0 3 3 3 2 1 2]*

*[0 3 3 3 0 2 1]*

*[0 3 3 3 2 0 2]*

*[0 3 2 3 2 0 0]*

*[0 3 3 3 2 2 0]*

*[0 3 3 3 2 2 2]*

*[0 2 2 3 3 3 2]] TIFF TIFF*

# Problem

Which functions listed use double nested python loops?

Fast Sub Values, CreateIList, FastNW

None of the listed functions use double nested for loops written in python. These python functions use modules that may include python code that has for loops.